Stabilizing the Richardson Algorithm by Controlling Chaos

Song He

Bell Laboratories, Lucent Technologies

Murray Hill, NJ 07974

`song@bell-lab.com`

(May 9, 1996)

Abstract

By viewing the operations of the Richardson purification algorithm as a discrete time dynamical process, we propose a method to overcome the instability of the algorithm by controlling chaos. We present theoretical analysis and numerical results on the behavior and performance of the stabilized algorithm.

Typeset using REVTEX

In 1950 Richardson [1] proposed an algorithm to construct the eigenvectors of a hermitian matrix when its eigenvalues are given. The basic idea of the method is very simple: start from an arbitrary initial vector, and use the eigen equation to eliminate the unwanted components in the initial vector. Specifically, to compute the eigenvector $|\varepsilon_k\rangle$, we apply the operations according to the equation

$$|\varepsilon_k\rangle \sim \prod_{i\neq k}(\mathbf{H} - \varepsilon_i)\,|\phi^{(0)}\rangle, \tag{1}$$

where $|\phi^{(0)}\rangle$ is an arbitrary initial vector which has finite overlap with $|\varepsilon_k\rangle$ and $\{\varepsilon_i\}$ is the set of eigenvalues of the $N \times N$ hermitian matrix $\mathbf{H}$. If the operations can be done to infinite precision, this procedure gives the desired eigenvector in $N-1$ iterations. However, it is known that this algorithm is unstable [2], *i.e.* an initial vector close to the desired eigenvector is driven away from it under the operations of the algorithm. To see this, let us consider a vector

$$|\phi\rangle = |\varepsilon_k\rangle + \sum_{i\neq k}\delta_i\,|\varepsilon_i\rangle\,, \tag{2}$$

where δ_i are small compared to 1. In the step of the algorithm when $(\mathbf{H} - \varepsilon_j)$ is applied on $|\phi\rangle$, the resultant vector is proportional to

$$|\phi'\rangle = |\varepsilon_k\rangle + \sum_{i\neq k}\delta_i'\,|\varepsilon_i\rangle\,, \tag{3}$$

where

$$\delta_i' = \delta_i\frac{\varepsilon_i - \varepsilon_j}{\varepsilon_k - \varepsilon_j}. \tag{4}$$

The above equations show that the fixed point $|\varepsilon_k\rangle$ of the map $(\mathbf{H} - \varepsilon_j)$ is unstable. The exponents characterizing the instability can be as large as $\ln W$, where W is the natural band width defined by the difference between the largest eigenvalue and the smallest eigenvalue devided by the smallest level spacing. If the natural band width W is large, then the algorithm is highly unstable. It was proposed [2] that one may make the algorithm work better in special cases by replacing Eqn.(1) by

$$|\varepsilon_k\rangle \sim \prod_{i \neq k} (\mathbf{H} - \varepsilon_i)^{n_i} |\phi^{(0)}\rangle \tag{5}$$

with $n_i \geq 1$. However, this does not overcome the instability of the original algorithm. Experimentation with this proposal showed that the scope of its applicability is very limited. Therefore the algorithm was concluded to have little practical use [2].

In this paper, we revisit the instability problem of the Richardson algorithm. By viewing the operations of the algorithm as a discrete time dynamical process and using ideas similar to controlling chaos [3–5], we are able to devise a method to stabilize the algorithm. The key observation is that *by dynamically arranging the order of the operations in the product* $\prod_{i \neq k} (\mathbf{H} - \varepsilon_i)^{n_i} |\phi^{(0)}\rangle$, *we can use the instability itself to enhance the amplitude of the desired eigenvector in the initial vector while suppressing those of the unwanted ones.* In the rest of the paper, we first present the stabilized algorithm. Following that we present a theoretical analysis of the behavior of the stabilized algorithm. Finally we present numerical results on the behavior and the the effectiveness of the stabilized algorithm: a randomly generated initial vector converges to the desired eigenvector exponentially on the average under the operations of the algorithm.

To map the operations under the algorithm to a discrete time dynamical process, we rewrite Eqn.(1) as an iteration:

$$|\phi^{(n+1)}\rangle = (\mathbf{H} - \varepsilon^{(n)}) |\phi^{(n)}\rangle, \tag{6}$$

where the sequence $\varepsilon^{(n)} \in \{\varepsilon_i, 1 \leq i \leq N, i \neq k\}$ defines the dynamics. As the discrete time n increases, the initial state vector $|\phi^{(0)}\rangle$ of the dynamical system is transformed to $|\phi^{(n)}\rangle$ by the set of matrices $\{\mathbf{H} - \varepsilon_i : 0 \leq i \leq N, i \neq k\}$. These matrices map the N-dimensional unit sphere to itself if the vectors are normalized, which is assumed through out our paper. Eqn.'s (3) and (4) show that the behavior of these maps around $|\varepsilon_k\rangle$ is very complex. There are large number of stable as well as unstable directions. The behavior of the system is further complicated by rounding errors on a finite precision computer. One of the consequences of a finite precision is that the operation in Eqn.(6) becomes nonlinear and the system becomes

chaotic. To see this, consider the operation of $\mathbf{H} - \varepsilon_l$ on a vector $|\phi\rangle$ on a finite precision computer. To be more precise, we define a function

$$\Re_{\varepsilon_l,\varsigma}(|\phi\rangle) \equiv (\mathbf{H} - \varepsilon_l)\,|\phi\rangle, \tag{7}$$

where ς is the precision of the computer, *i.e.* for any $x \in (-\varsigma, \varsigma)$, the computer gives $1+x = 1$. Let $|\phi\rangle = |\varepsilon_i\rangle + \gamma\,|\varepsilon_j\rangle$ and γ be sufficiently small. We have on the one hand

$$\Re_{\varepsilon_l,\varsigma}(|\phi\rangle) = \Re_{\varepsilon_l,\varsigma}(|\varepsilon_i\rangle) = (\varepsilon_i - \varepsilon_l)\,|\varepsilon_i\rangle\,. \tag{8}$$

On the other hand,

$$\Re_{\varepsilon_l,\varsigma}(|\varepsilon_i\rangle) + \gamma\Re_{\varepsilon_l,\varsigma}(|\varepsilon_j\rangle) = (\varepsilon_i - \varepsilon_l)\{|\varepsilon_i\rangle + \gamma'\,|\varepsilon_j\rangle\}, \tag{9}$$

where

$$\gamma' = \gamma\frac{\varepsilon_j - \varepsilon_l}{\varepsilon_i - \varepsilon_l}. \tag{10}$$

Clearly if the natural band width W is large enough, the term proportional to γ' on the right hand side of Eqn.(9) may not be negligible. Therefore the function $\Re_{\varepsilon_l,\varsigma}(|\phi\rangle)$ is *nonlinear*. We emphasize that the nonlinearity of $\Re_{\varepsilon_l,\varsigma}(|\phi\rangle)$ also makes the order of the operations in Eqn.(1) or Eqn.(5) important. In Fig.1, we show the typical behavior of the system described by Eqn.(6). The lower panel shows the separation $\zeta = \|\,|\phi_1\rangle - |\phi_2\rangle\|$ between two initially neighboring vectors as a function of the discrete time. The initial separation between these two vectors is about 10^{-16}. The upper panel shows the computed Lyapunov exponent λ, which converges to about 0.23. The matrix used to obtain these results is a 4096×4096 tridiagonal matrix, which is more precisely defined later in the paper. We choose k to be 1. The sequence $\{\varepsilon^{(n)}\}$ which determines the dynamics is periodic and is given by $\varepsilon^{(n)} = \varepsilon_{P_j}$, where $j = n \bmod (N - 1)$ and P is a random permutation of the $N - 1$ indices $\{2, \cdots, N\}$. The irregular behavior of ζ in the lower panel of the figure is not due to any randomness. It is due to the deterministic chaotic dynamics defined by Eqn.(6) on a finite precision computer. The task of stabilizing the algorithm is to determine $\varepsilon^{(n)}$ dynamically

4

from the set of *discrete* values $\{\varepsilon_i, i \neq k\}$ so that $|\phi^{(n)}\rangle$ converges to $|\varepsilon_k\rangle$ as n increases. The ideas of controlling chaos come into the construction of the sequence $\{\varepsilon^{(n)}\}$. It is interesting to contrast our case with that of the usual problem of controlling chaos [4,5] where a set of *continuous* parameters is adjusted in dragging the system back to the desired periodic orbit.

The iterative procedure to compute the eigenvector $|\varepsilon_k\rangle$ using the stabilized algorithm is the following.

A-1 Generate a normalized random initial vector $|\phi^{(0)}\rangle$ and an random array $\{a_i^{(0)} : 0 < a_i^{(0)} < 1\}$.

A-2 In the n'th step of the algorithm, the parameter $\varepsilon^{(n)}$ is determined by $\varepsilon^{(n)} = \varepsilon_j$, where j is such that $a_j^{(n)} = \max_{i \neq k}\{a_i^{(n)}\}$. The *normalized* vectors $|\phi^{(n)}\rangle$ and the array $\{a_i^{(n)}\}$ are updated according to the equations

$$|\phi^{(n+1)}\rangle = (\mathbf{H} - \varepsilon^{(n)})\,|\phi^{(n)}\rangle, \tag{11}$$

$$a_i^{(n+1)} = \begin{cases} \frac{\bar{\delta}}{\Delta} & i = j, \\ \frac{a_i^{(n)}|\varepsilon_i - \varepsilon^{(n)}|}{\Delta} & i \neq j, \end{cases} \tag{12}$$

where Δ is the smallest level spacing, $\bar{\delta} < \Delta$ is a small number comparable to the accuracy δ (defined in Eqn.(19)) to which the eigen equation is satisfied.

A-3 Terminate the iteration if the error parameter $\bar{\sigma} = \{\frac{1}{N} \| (\mathbf{H} - \varepsilon_k)\,|\phi^{(n)}\rangle \|^2\}^{\frac{1}{2}}$ is smaller than required.

Let M be the total number of iterations and m_i be the number of occurrences of ε_i in the sequence $\{\varepsilon^{(n)}\}$. Clearly $m_k = 0$. Less obviously $m_i \propto M$ as $M \longrightarrow \infty$. The latter relation comes about for the following reason: A newly eliminated element in the array $\{a_i^{(n)}\}$ is set to a small residual value $\bar{\delta}/\Delta$ in the iteration. It grows exponentially in the later iterations and is chosen to be eliminated again by the algorithm.

To analyze the behavior of the algorithm, we need to obtain the accumulated effect of the iterations on the initial vector. For that purpose, we write

5

$$| \phi^{(0)} \rangle = \sum_i c_i^{(0)} | \varepsilon_i \rangle \,, \tag{13}$$

where $| \varepsilon_i \rangle$ are the eigenvectors of the matrix $\mathbf{H}$. On a finite precision computer, the eigen equation is only satisfied to about the machine precision, *i.e.* we have

$$(\mathbf{H} - \varepsilon_i) | \varepsilon_i \rangle = \sum_j R_{i,j} | \varepsilon_j \rangle \,, \tag{14}$$

where $R_{i,j}$ is the same order as the machine precision ς. The resultant vector is

$$| \phi^{(M)} \rangle \propto \prod_n (\mathbf{H} - \varepsilon^{(n)}) \cdot \sum_i c_i^{(0)} | \varepsilon_i \rangle = \sum_i c_i^{(M)} | \varepsilon_i \rangle \,, \tag{15}$$

where

$$c_i^{(M)} = c_i^{(0)} \prod_n (\varepsilon_i - \varepsilon^{(n)}) = c_i^{(0)} \prod_{j \neq i} (\varepsilon_i - \varepsilon_j)^{m_j} R_{i,i}^{m_i}, \tag{16}$$

and the terms higher order in ς have been ignored. To show that $| \phi^{(M)} \rangle$ indeed converges to $| \varepsilon_k \rangle$, we compute the ratio

$$\left| \frac{c_i^{(M)}}{c_k^{(M)}} \right| \leq \left| \frac{c_i^{(0)}}{c_k^{(0)}} \right| \prod_{j \neq i,k} \left| \frac{\varepsilon_i - \varepsilon_j}{\varepsilon_k - \varepsilon_j} \right|^{m_j} \left| \frac{\delta}{\varepsilon_k - \varepsilon_i} \right|^{m_i} \tag{17}$$

$$\equiv \left| \frac{c_i^{(0)}}{c_k^{(0)}} \right| r_{i,k}^{(M)}, \tag{18}$$

where the parameter δ is defined by

$$\delta = \max_{i \neq k} \{ |R_{i,i}| \}. \tag{19}$$

We remark that: (1) If $\left| \frac{\delta}{\varepsilon_k - \varepsilon_i} \right| \ll 1$, which usually is the case in practical applications, the vanishing ratio $r_{i,k}^{(M)}$ is dominated by $\left| \frac{\delta}{\varepsilon_k - \varepsilon_i} \right|^{m_i}$. The initial values of $\{ c_i^{(0)} \}$ do not affect the convergence of the algorithm. (2) The motivation of introducing the array $\{ a_i^{(0)} \}$ in the stabilized algorithm is to replace the *a priori* unknown coefficients $\left| c_i^{(0)} \right|$ by $a_i^{(0)}$. The previous comment justifies our so doing.

To demonstrate the practical effectiveness and to illustrate the numerical behavior of our algorithm, we show two sets of numerical results. In the first example, we deal with computing the eigenvectors of of a tridiagonal matrix after its eigenvalues have been obtained

by the usual implicit QL method [2,6]. In the second example, we deal with the problem of computing the basis vectors in the decomposition of a direct product of irreducible representations of the $SU(2)$ algebra. In this case, the algorithm is especially effective because the eigenvalues of the matrix involved, the total angular momentum operator, are known exactly due to the algebraic properties of $SU(2)$.

In the first example, we demonstrate the behavior and the performance of the stabilized algorithm by computing the eigenvectors of a tridiagonal matrix of dimension $N = 4096$. The diagonal terms $H_{i,i} \in [-1, 1]$ are randomly generated with a uniform distribution. Its next diagonal terms are all 1. The eigenvalues of the matrix are obtained by the standard implicit QL procedure. For the particular matrix we used, the smallest level spacing Δ equals 1.67×10^{-6} and the natural band width W equals 2.7×10^6. The parameter $\bar{\delta}$ should be the same order as the machine precision, which is usually 10^{-13} with double precision. In our calculations, we choose it to be $\bar{\delta} = 10^{-10}$. We will comment on the effect of the exact magnitude of $\bar{\delta}$ on the behavior of the algorithm later in the paper.

In Fig.2 and Fig.3, we show respectively the error parameter σ defined by

$$\sigma = \{\frac{1}{N} \| \, | \phi^{(n)} \rangle - | \varepsilon_k \rangle \, \|^2\}^{\frac{1}{2}} \tag{20}$$

as a function of n for $k = 1$, the lowest eigenvalue, and $k = 3097$, the eigenvalue next to the smallest level spacing Δ. For the eigenvector $|\varepsilon_k\rangle$ in the above equation, we use the converged eigenvector obtained from our stabilized algorithm according to A-1, A-2, and A-3. For comparison, the results for σ calculated from a naive implementation of algorithm without controlling chaos are also shown as dashed lines in the figures. In the naive implementation of the algorithm, the unwanted components of the initial vector are eliminated periodically instead of according to the procedure outlined in A-1, A-2, and A-3. The effectiveness of the stabilized algorithm is evident: the error parameter σ decays exponentially on the average as a function of n after an initial transient period. In particular, for $k = 1$, we see that the initial vector converges to the desired eigenvector within machine precision in about $M = 300$ iterations. This is much fewer than the number of iterations $N - 1 = 4095$

required by an implementation of the original algorithm with infinite precision. This shows that by dynamically arranging the order of the operations, we can indeed use the instability itself to enhance the desired component in the initial vector while suppressing the unwanted ones. For $k = 3097$, the initial vector manages to converge to the desired eigenvector within machine precision in about $1.5 \times 10^4 \sim 4N$ iterations. We find that this eigenvector is the most difficult to compute because its eigenvalue is next to the smallest level spacing Δ. This is similar to other algorithms [2] for computing eigenvectors. We also notice that σ exhibits large fluctuations as a function of n, even though its magnitude always remains at around 10^{-10}. Detailed analysis of the operations of the algorithm indicates that these fluctuations are due to the instability of the original algorithm. However, we want to emphasize that these fluctuations do not affect the computation of the desired eigenvector since we can always terminate the iteration once σ is smaller than required.

In Fig.4, we show the quantity $r_{i,k}^{(M)}$ defined in Eqn.(17) as a function of i for $k = 3097$. In the inset, we show m_i as a function of i. The similarity between the curves in the figure show that $r_{i,j}^{(M)}$ is indeed dominated by the contribution from $\left| \frac{\delta}{\varepsilon_k - \varepsilon_i} \right|^{m_i}$, as asserted above in the analysis of the algorithm. The largest magnitude of $r_{i,k}^{(M)}$ is very small, about 10^{-25}. This guarantees the convergence to the desired eigenvector for almost any initial vector $| \phi^{(0)} \rangle$. We also find that the relative magnitude of m_i depends on the local level spacing around the eigenvalue ε_i. Our calculations indicate that m_i is large (small) when the local level spacing is large (small).

We have experimented with our algorithm on variety of matrices with different eigen spectra structure. Our experience indicates that convergence is maintained even when δ/Δ is comparable to 1. We find that the number of iterations M needed for the algorithm to compute a converged eigenvector of the low-lying eigenvalues within machine precision does not increase much as N increases for any given class of matrices. This means that for these eigenvectors our algorithm converges in about αN^2 operation, where α depends on the eigen structure of the matrices and is a very slowly varying function of N. In the case of tridiagonal matrices with random diagonal elements and for the eigenvector of the lowest eigenvalue,

our calculations on matrices of dimensions from 512 to 131072 indicate that α is basically a constant around 300. For a typical eigenvector, the algorithm converges in about $M = \beta N$ iterations, $i.e.$ βN^3 operations, where β is a number comparable to 1. The exactly value of β depends on the local level spacing. We find that the parameter $\bar{\delta}$ used in the algorithm has effects similar to that of the temperature in an optimization problem by simulated annealing. When $\bar{\delta}/\Delta$ is small, we find that the vector $\mid \phi^{(n)} \rangle$ may be trapped around a particular vector for a long time before it manages to escape. On the other hand, if $\bar{\delta}/\Delta$ is large, we find that it takes longer for the initial vector to converge and the error parameter σ fluctuates significantly as a function of n. Our experience also indicates that refreshing (replacing $a_i^{(n)}$ by its reciprocal, for example) the array $\{a_i^{(n)}\}$ periodically when σ has become small, 10^{-5} for example, helps to speed up the convergence. We believe this eliminates the bias built into the relative values of $\{a_i^{(n)}\}$ according to Eqn.(17) during the iterations. In comparison to other traditional methods of computing eigenvectors, the stabilized algorithm has a number of advantages. Here we want to compare it to the inverse iteration method, the best traditional method of computing eigenvectors when the eigenvalues are known [2]. It typically takes about N^3 operations for inverse iteration to compute an eigenvector. This is comparable to the number of operations needed by our algorithm in the worst cases. However, the inverse iteration method needs more computer memory because it needs to store the inverse of the matrix $\mathbf{H}-\varepsilon_k$. If the matrix $\mathbf{H}$ is unstructurally sparse, it is clear that our algorithm can take advantage of the sparsity but the inverse iteration method cannot because the inverse of $\mathbf{H} - \varepsilon_k$ may not be a sparse matrix.

In the second example, we use the algorithm to construct the basis vectors in the decomposition of a direct product of irreducible representations of the algebra $SU(2)$. The specific case we studied is to construct the basis vectors of the irreducible representations contained in the product $\underbrace{S \otimes S \otimes \cdots \otimes S}_{N}$. This is a generalization of the usual Clebsch-Gordan coefficients used to couple two angular momenta together. In our example, what we want to do physically is to construct the eigenstates of the total angular momentum

operator $\mathbf{L}$ of N spinless electrons with orbital angular momentum S. This is very useful in block diagonalizing the Hamiltonian of a many electron system [7]. To be more specific, we choose $N = 8$ and $S = \frac{21}{2}$. From the algebraic properties of $SU(2)$, we know that $\underbrace{\frac{21}{2} \otimes \frac{21}{2} \otimes \cdots \otimes \frac{21}{2}}_{8} = 0 \oplus 1 \oplus \cdots \oplus 56$ for identical fermions. Therefore the eigenvalues of the total angular momentum operator $\mathbf{L}^2$ are *known* exactly: $\{\varepsilon_l = l(l+1), 0 \le l \le 56\}$. When the z component of the angular momentum is constrained to be $\mathbf{L}_z = 0$, the dimension of the matrix $\mathbf{L}^2$ is $D = 8512$. This is much larger than the number of distinct eigenvalues. Therefore the eigenvalues are highly degenerate. To construct the basis vectors of the eigen subspace Ω_l of dimension D_l, we simply start from D_l randomly generated initial vectors $\{|\phi_l^{(0)}\rangle, l = 1, 2, \cdots, D_l\}$ and apply our stabilized algorithm. It is easy to show that the final eigenvectors are linearly independent with probability 1.

In Fig.(5), we show the error parameter σ as a function of n for $l = 0$. Since the eigenvalues are massively degenerate, we have to modify the definition of σ to

$$\sigma = \{\frac{1}{N}\| \, |\phi^{(n)}\rangle - \sum_{i=1}^{D_l} |e_l^{(i)}\rangle\langle e_l^{(i)} \, |\, \phi^{(n)}\rangle\|^2\}^{\frac{1}{2}}, \tag{21}$$

where $\{|e_l^{(i)}\rangle\}$ is the known basis set of the eigen subspace Ω_l. The data shown in Fig.(5) are obtained for $l = 0$. The dimension of the eigen subspace is $D_0 = 31$. The dashed line shows σ obtained from a naive implementation of the original algorithm defined in previous discussions. We see that a randomly generated initial vector converges to the desired eigenvector within machine precision in about $M = 75$ iterations. We note that fluctuations in σ are also present, similar to the first example. We have studied the behavior of the algorithm with other values of l, N, and S where the dimension of the matrix $\mathbf{L}^2$ can be as large as 10^6. We find that in all cases, the initial vector converges to the desired eigenvector within machine precision in a number of iterations fewer than twice of the number of distinct eigenvalues of the operator $\mathbf{L}^2$. We conjecture that this is true for all values of l, N, and S. A useful point to note in speeding up the construction of the basis vectors is that we may use the same sequence $\{\varepsilon^{(n)}\}$ for the construction of all of the basis vectors. This avoids the computation efforts in constructing $\{\varepsilon^{(n)}\}$ for each of the initial vectors. One can

show that the basis vectors generated this way are linearly independent with probability 1 due to the instability.

We expect the stabilized algorithm to have similar advantages for other compact Lie algebras. The reasons are mainly twofold. (1) The algebraic properties of the Lie algebra involved usually determine the eigen spectra of the operators involved. The eigenvalues are known exactly and do not need to be computed. (2) The dimensions of the irreducible representations are usually large. This means that the eigen spectra of the operators involved are highly degenerate. Therefore the number of distinct eigenvalues and the natural band widths are small. Because of these features, we find the stabilized algorithm to be a very powerful tool in fully utilizing the symmetry of the physical system concerned [7].

Before concluding, we remark that the algorithm should work for general non-hermitian matrices as well. Depending on whether the desired eigenvector is a *left* or *right* eigenvector, we need to modify the operations of the matrix on the vector $|\phi^{(n)}\rangle$ to *left* or *right* multiplications. As a general remark, we find the idea of viewing the operations of an algorithm as a dynamical process very intriguing. It seems reasonable to believe that the explorations in this direction should be interesting and fruitful.

In summary, we have discussed a method to stabilize the Richardson purification algorithm using controlling chaos. We have presented theoretical analysis and numerical results to illustrate the behavior and the effectiveness of the stabilized algorithm. We find that by dynamically arranging the order of operations in the originally unstable algorithm, we are able to use the instability to enhance the desired components of a initial vector while suppressing the unwanted ones so that the initial vector converges to the desired eigenvector exponentially.

REFERENCES

[1] L. F. Richardson, *Phil. Trans. Roy. Soc.* A242, 439(1950).

[2] J. H. Wilkinson, "*The Algebraic Eigenvalue Problem*", Clarendon Press, Oxford, 1965.

[3] A. Hubler, *Helv. Phys. Acta* **62**, 343(1989); T. B. Fowler, *IEEE Trans. Autom. Control* **34**, 201,(1989).

[4] E. Ott, Celso Grebogi, and J. A. Yorke, *Phys. Rev. Lett.* 64, 1196(1990).

[5] D. Auerbach, C. Grebogi, E. Ott and J. A. Yorke, *Phys. Rev. Lett.* 69, 3479(1992); U. Dressler and G. Nitsche, *Phys. Rev. Lett.* 68, 1(1991); Hu Guang and Qu Zhilin, *Phys. Rev. Lett.* 72, 68(1994); R. Roy, T. W. Murphy, Jr., T. D. Maier, and Z. Gills, *Phys. Rev. Lett.* 68, 1259(1992).

[6] B. T. Smith, *et al.* "*Matrix Eigensystem Routines–Eispack guide*", Springer-Verlag, New York, 1976.

[7] Song He, S. H. Simon, and B. I. Halperin, *Phys. Rev.* **B**50, 1823(1994).

FIGURES

FIG. 1. Generic behavior of the system. Upper panel: Lyapunov exponent. Lower panel: separation between two initially neighboring vectors.

FIG. 2. The error parameter σ as a function of the number of iterations for the lowest eigenvalue. Solid line: with controlling chaos. Dashed line: without controlling chaos.

FIG. 3. The error parameter σ as a function of the number of iterations for the eigenvalue next to the smallest level spacing Δ. Solid line: with controlling chaos. Dashed line: without controlling chaos.

FIG. 4. $\log_{10} r_{i,k}^{(M)}$ as a function of the eigenvalue index i for $k = 3096$. Inset shows m_i as a function of i.

FIG. 5. The error parameter σ as a function of the number of iterations for $l = 0$. Solid line: with controlling chaos. Dashed line: without controlling chaos.